Rule-based Modeling and Simulation of Biochemical Systems with Molecular Finite Automata

Jin Yang[*], Xin Meng[1], William S. Hlavacek[†‡]

Abstract

We propose a theoretical formalism, molecular finite automata (MFA), to describe individual proteins as rule-based computing machines. The MFA formalism provides a framework for modeling individual protein behaviors and systems-level dynamics via construction of programmable and executable machines. Models specified within this formalism explicitly represent the context-sensitive dynamics of individual proteins driven by external inputs and represent protein-protein interactions as synchronized machine reconfigurations. Both deterministic and stochastic simulations can be applied to quantitatively compute the dynamics of MFA models. We apply the MFA formalism to model and simulate a simple example of a signal transduction system that involves a MAP kinase cascade and a scaffold protein.

Keywords: Rule-based modeling, executable biology, finite state machine, computational systems biology, formal languages, cell signaling

1 Introduction

In computational systems biology, studying a complex biochemical system involving a large number of interacting proteins often relies on *in silico* simulations to analyze and predict system behaviors [1]. In recent years, computational models have been increasingly used in cell signaling research and have been developed to study various pathways [2, 3]. However, models often fail to capture the mechanistic details of signal transduction systems [4]. For example, models sometimes inadequately account for the complexities of protein interactions, including interaction details at the level of protein sites and structural relationships among components of signaling proteins [5], particularly multisite protein modification in the context of multiprotein complexation [6]. Proteins

[*]Chinese Academy of Sciences – Max Plank Society Partner Institute for Computational Biology, Shanghai Institutes for Biological Sciences, Shanghai 200031, China. E-mail: jinyang2004@gmail.com

[†]Theoretical Division and Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

[‡]Department of Biology, University of New Mexico, Albuquerque, NM 87131, USA

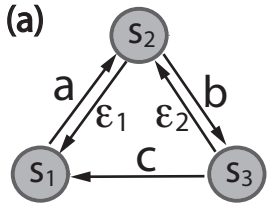
in a signal-transduction system often have multiple component parts that enable the protein to interact with other molecules in a modular manner [7, 8, 9]. Models that account for the functions of the component parts of proteins (e.g, linear motifs and protein interaction domains) are needed to better understand the dynamics of signal-transduction systems [10, 11].

Limitations of conventional modeling approaches, which rely on explicit specifications of chemical reaction networks, lie in both model construction and simulation. Conventional models are essentially specified with lists of biochemical species and their reactions. However, representing a biochemical system as a chemical reaction network is often cumbersome and unnecessary [12, 13]. Graphical rule-based modeling formalisms and associated simulation algorithms have been developed to represent biochemical systems in terms of formal rules for biomolecular interactions [12, 13, 14, 15, 16, 17, 18, 19]. In graphical rule-based modeling, graphs (or the equivalents) are used to represent molecules, and graph-rewriting rules (or the equivalents) are used to represent molecular interactions. A rule represents a molecular interaction explicitly and the reactions that can arise from the interaction implicitly, and a rule can be viewed as a coarse-grained description of a class of reactions.

Two common types of protein interactions, multivalent protein binding and multisite post-translational protein modification, cause a combinatorial increase in the size of a reaction network with an increase in the number of interaction modules. It is usually difficult and error prone to manually construct a full-sized chemical reaction model. As an alternative, such conventional models can be automatically obtained using rule-based reaction generation tools, such as BioNetGen, Virtual Cell or little b [20, 21, 22, 23], Moleculizer or Smoldyn [24, 25], Simmune [26], or Stochastic Simulator Compiler (SSC) [27]. Unfortunately, the number of reactions and biochemical species implied by rules can be enormously large (even infinite or limited only by the number of molecules in a system) for rule-based models of signal transduction systems [12], making it inefficient to construct, simulate and analyze conventional models derived from rules.

In addition to formalisms based on graph rewriting, a number of theoretical frameworks for biomolecular interaction systems have been proposed over the past decade or so to facilitate model building and simulation. Despite the differences in their syntactical and grammatical structures, most formalisms share a common feature: molecular entities are treated as computational agents that interact with one another according to a collection of specific protocols [28, 29, 30, 31, 32]. For example, protein interactions have been viewed as concurrent processes and have been modeled with communication protocols by process algebras, such as π-calculus [29, 30]. Many of these formalisms have been coupled to Gillespie's stochastic simulation algorithm [30, 33] to enable discrete-event simulation.

In engineering and computer science, complex dynamical systems with heterogeneous, modular and reactive components are frequently modeled by state machines and related formal structures. In this paper, we propose a new formalism, referred to as *molecular finite automata* (MFA), to model individual proteins as structured computing agents and to specify protein-protein interactions in the form of synchronized dynamics of interacting agents. The main goal is to provide an intuitive as well as programmable representation for biomolecular interaction systems. The MFA formalism is developed by incorporating and extending the classic structure of finite automata, which is a well-established



(b)

	a	b	c	ε_1	ε_2
s_1	s_2	–	–	–	–
s_2	–	s_3	–	s_1	–
s_3	–	–	s_1	–	s_2

Figure 1: An example finite automaton. (a) State transition diagram for a three-state finite automaton. A circle denotes a state. An arrow denotes a state transition. A letter next to an arrow denotes an input. (b) A state transition table for the finite automaton in panel (a). The leftmost column indicates possible current states. The topmost row indicates inputs. A table entry indicates a target state given a current state and an input. The symbol '–' indicates "not applicable."

formalism that has a wide application range and for which many sophisticated software and hardware tools are available. As we will see, an agent within the MFA framework explicitly represents a protein's activity (or state) induced by external inputs, and a protein interaction is specified as a joint transformation of the states of multiple MFA agents. At the systems level, a collection of *reaction rules* is used to describe interactions among the MFA agents in a system.

We also report simulation methods that can compute the dynamics of a system modeled within the MFA framework. Using the example of a MAP kinase cascade, we demonstrate how to apply the MFA formalism to model and simulate a cell signaling system.

2 Formal model

In the first part of this section, we introduce a representational framework using MFAs to describe the building blocks, particularly proteins, of biomolecular interaction systems. In the second part, we show how to apply the MFA representations to construct quantitative models for cell signaling systems and how to compute with these models.

2.1 Molecular entities — molecular finite automata

The notion of finite automata, or finite state machines, is well-established in theoretical computer science and has been applied to model the dynamics of diverse discrete systems (i.e., systems with finite numbers of states) [34]. Finite automata were traditionally developed to construct parsers and compilers, conduct formal verifications and mathematical proofs, and design and test software [34, 35, 36]. Because of their simple, adaptive and intuitive structure, finite automata have been applied in a number of areas, including engineering systems design [37], computational linguistics [38] and communication protocols [39]. Interacting state machines have also been used in the field of computational biology to visualize and model cellular level interactions [32].

Our goal in this paper is to propose a formalism based on an extended structure of finite automata that is suited for modeling biomolecular intreactions at

the submolecular level with consideration of site-specific details. Below, we give a formal definition of a purely reactive finite automaton that will be extended for the later description of biomolecules.

Definition 1 (finite automaton). *A finite automaton is a tuple $D = (S, X, \delta, s_0)$, where S and X are finite sets of states and inputs, respectively. The function δ is a transition function that maps the current state along with an input in X into a target state, $\delta : S \times X \to S$. The symbol s_0 denotes a start state.*

The automaton of Definition 1 is a so-called "deterministic" finite automaton (DFA). In a DFA, given an input, a state transition is non-ambiguous, and a DFA can only reside in one state at any given time. In contrast, a finite automaton can be nondeterministic, in which multiple state transition paths (or more than one target state) may exist for a single input. Here, assuming responses of a protein to external signals are deterministic, we focus on DFAs as the fundamental structures for modeling proteins.

The above definition characterizes a finite automaton as a reactive model in which state transitions are induced by external events in the form of input signals. Elements in the set of states S are represented using subscripted lower-case letters, $S = \{s_1, s_2, ...,\}$. Throughout, inputs are represented using the lower-case alphabet, ε and subscripted ε's, i.e., $X = \{a, b, ..., \varepsilon, \varepsilon_1, \varepsilon_2, ...\}$. A special input symbol ε is introduced to model a non-specific external signal or a signal from an unknown source that causes a spontaneous state transition. In a model of a signaling system, such a non-specific input can be used to model molecular events such as dissociation of two bound proteins caused by background collisions with solvent molecules or protein modifications catalyzed by unknown enzymes. A finite automaton can be visually represented by a *state transition diagram* (Fig. 1(a)), a directed graph in which a node denotes a state and an edge denotes an input-induced transition. Equivalently, a finite automaton can be specified by a machine-readable *state transition table* (Fig. 1(b)).

The dynamics of many reactive systems can be represented using the DFA structure of Definition 1. However, this structure is inefficient for describing signaling proteins. To extend the DFA structure to model a protein, we first look at the correspondence between properties of finite automata and protein functions. A classic finite automaton models a memoryless process, wherein a state transition depends only on the current state and an input. In contrast, protein interactions mostly happen under certain molecular contexts. To see the importance of molecular context, we consider allosteric regulation and protein complexation. Allosteric regulation of a protein or enzyme is a common mechanism in biochemistry. Protein activity in one domain is changed (either activated or inhibited) by binding or unbinding of an effector molecule at another site. The formation of heterogeneous and transient multiprotein complexes is one of the essential functions of protein-protein interactions in signal transduction. Context-sensitive interactions such as co-localization of an enzyme and one of its substrates control both the strength and specificity of molecular signaling. These features of protein interactions require an extension of the DFA structure beyond the representation of information only in terms of a finite number of states.

To capture the contextual sensitivity of protein interactions, internal variables are introduced to record contextual information such as information about binding partners or other local molecular information. An extension of Definition 1 should also include functions that will read and modify the machine

Table 1: Operators and symbols used in MFA structures

Operator/symbol	Definition
$x := a$	Assignment of a value a to a variable x
$x = a$	Comparison between x and a
$/a$	Delimiter that precedes an operation a
$\backslash a$	Delimiter that precedes a predicate a
$a.b$	Component operator: b is a member of a
A–B	Bond association between A and B
$x \rightarrow A$	Mapping input x to machine A

variables. Along with state transitions, these machine operations update the configuration of a protein. Based on these considerations, we define an enhanced automaton structure, an "extended finite automaton" (EFA) to amend the classic finite automata structure.

Definition 2 (extended finite automaton). *An extended finite automaton is a tuple* $E = (S, X, \delta, s_0, \mathbf{v})$*, where* S *and* X *are finite sets of states and inputs, respectively. The transition function,* $\delta : S \times X \backslash P(\mathbf{v}) \rightarrow S/A(\mathbf{v})$*, maps the current state along with an input in* X *into a target state upon evaluation of a predicate function* $P(\mathbf{v})$*, and performs an operation* $A(\mathbf{v})$ *on the variable structure* $\mathbf{v}$ *along with the state transition. The symbol* s_0 *denotes a start state.*

The meanings of operators (e.g., \ and /) used in the above definition are given in Table 1. Our definition of EFA is close to the convention of an extended finite state machine [36], which also involves operations on internal variables. Suppose that an EFA E is in state s. Upon receiving an input x, E undergoes a transition $\delta = (s, q, x, P(\mathbf{v}), A(\mathbf{v}))$, where s and q are the source and target states, respectively. If the predicate $P(\mathbf{v})$ is true (e.g., an evaluation of variables in $\mathbf{v}$ indicates that the transition is legitimate), E moves to the target state q and performs an operation $A(\mathbf{v})$ on the variable structure $\mathbf{v}$.

Ultimately, another important and ubiquitous feature of cell signaling, site-specific interactions, must be incorporated to reflect the modularity of protein interactions. Many signaling proteins possess multiple functional motifs, domains and sites, which serve as modules for combinatoric protein organizations that can potentially generate diverse signaling patterns. A realistic protein automaton should express dynamics at the level of protein sites. To this end, we arrive at the definition of "molecular finite automaton" (MFA), which models the discrete dynamics of a multidomain biomolecule. The relationship between MFA and EFA is as follows: (1) an MFA contains one or multiple EFAs and (2) each EFA in an MFA operates on a common variable structure that is shared by all EFAs. Table 2 summarizes a conceptual mapping between protein functions and the structure of an MFA.

Definition 3 (molecular finite automaton). *A molecular finite automaton is a tuple* $M = (E_1, E_2, ..., E_n, \mathbf{v})$*, which is composed of* n *component EFAs and a shared variable structure* $\mathbf{v}$*. The transition function for* E_i *is* $\delta_i : S_i \times X_i \backslash P_i(\mathbf{v}) \rightarrow S_i/A_i(\mathbf{v})$*.*

Table 1 lists a set of operators and symbols that we will use to describe MFAs in state transition diagrams and state transition tables. In essence, the MFA structure encapsulates multiple finite automata and allows for a hierarchical description of the component substructures of proteins. Internal variables

Table 2: Molecular finite automaton and protein function

MFA component	**Protein**
State	Conformation
State transition	Conformation change
Input	Biochemical interaction
Variable and predicate	Molecular context
Component machine	Domain or site

and predicates help to compress the state space and make an MFA more accessible to intuitive understanding. Without using internal variables and predicate functions, one can still build an MFA by expanding the state space assuming that variables store information of finite size. However, such an approach may result in a state expansion that might become intractable for a complex system.

The construction of an MFA requires knowledge and/or a hypothesis about the biochemistry and the component substructure of the protein one wants to model. Although some proteins have established functions in well-studied signaling pathways, biochemical mechanisms for many protein functions still await characterization. For a protein with known structure and function, the corresponding MFA must be designed to faithfully reproduce the reactive dynamics of the protein. For a poorly characterized protein, building an MFA, as in building any model, provides an opportunity to generate testable hypotheses.

As a design issue, MFA models can be constructed with a great deal of flexibility. Equivalent MFAs may differ in the number of states and the topology of state transition diagrams. A protein with multiple sites can be modeled by an MFA that has separate finite automata, each of which describes the dynamics of a domain. Equivalently, instead of using one automaton to model one protein site, the protein can be modeled by an MFA that consists of a single finite automaton that describes the combined behavior of all sites. For example, if a biomolecule has three independent domains that interact with different binding partners, it can be modeled as one eight-state finite automaton plus a variable structure (Fig. 2(a)), where state s_1 indicates that the molecule is in a free form with no binding partners and state s_8 indicates that all sites are occupied. Alternatively, the biomolecule can be modeled with three two-state (a free state and a bound state) EFAs with each EFA describing an individual binding domain (Fig. 2(b)). For the case of three identical and non-cooperative binding domains, it may be preferable to model the protein with a four-state finite automaton with the state space $S = \{s_1 : \text{free}, s_2 : \text{singly-bound}, s_3 : \text{doubly-bound}, s_4 : \text{triply-bound}\}$ for a parsimonious structure in terms of the number of states (Fig. 2(c)). This four-state MFA can be further compressed to a two-state model as shown in Fig. 2(d), where s_1 denotes the unoccupied state and s_1 denotes the protein is occupied at least on one of its three sites. In this model, the information about how many sites are bound is resolved by a variable c serving as a counter.

The state space of an MFA, S_M, is a subset of the product of the state spaces of component EFAs, i.e., $S_M \subseteq S_1 \times S_2 \times ... \times S_n$, where the two sides achieve equality when all component EFAs are independent. The input set of an MFA is a union $X_M = \bigcup_{i=1}^{n} X_i$. For an input $x \in X_M$, the transition function δ_i is chosen if x only belongs to X_i. A transition function is chosen arbitrarily if x

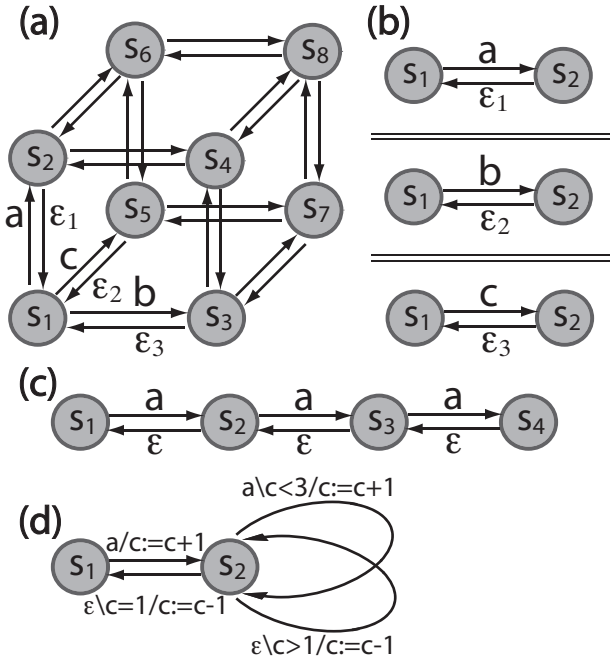


Figure 2: A protein with three binding domains modeled by different MFA structures. (a) An MFA that uses a single eight-state EFA to model the overall state transitions. Unlabeled transitions are induced by the same inputs as those identified for parallel transitions. (b) An MFA that models the protein with three independent internal finite automata, each of which interacts with distinct binding partners implied by input symbols $\{a, b, c\}$. Spontaneous inputs (ε_1, ε_2 and ε_3) are distinguished for non-identical individual EFAs. (c) A four-state MFA that models the protein as having three independent and identical binding sites. Machine variables that register binding partners are not shown. (d) A two-state MFA that can replace the model in (c). The two states s_1 and s_2 represent "free" and "bound", and the variable c counts the number of bound sites.

belongs to input sets of multiple component EFAs. For example, if $x \in X_i \cap X_j$, either δ_i from E_i or δ_j from E_j can be equivalently chosen to react to the input x. This scenario of relaying inputs corresponds to the case where a protein has multiple domains that interact with identical partners.

To present a biological example, Fig. 3 shows the state-transition diagram of an MFA model of the high-affinity IgE receptor, FcϵRI, in the model of Goldstein et al. [40] and Faeder et al. [41]. The state transition table for the MFA representation of FcϵRI is shown in Table 3. The receptor molecule FcϵRI has three functional domains: (1) an extracellular α subunit responsible for binding its ligand, IgE (in fact, an IgE dimer is considered in the models in Refs. [40, 41]); (2) an intracellular β subunit that constitutively binds to Src-family protein tyrosine kinase Lyn when it is unphosphorylated and recruits Lyn with higher affinity upon phosphorylation; and (3) an intracellular γ subunit that recruits another protein tyrosine kinase Syk upon phosphorylation.

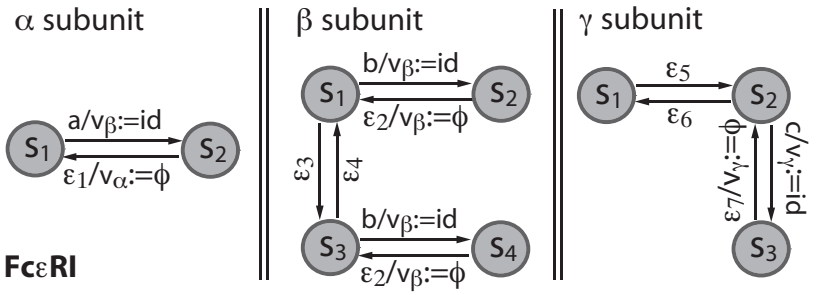


Figure 3: State transition diagram for the MFA of a receptor FcεRI with three component EFAs. α subunit: unbound (s_1), bound (s_2); β subunit: unbound and unphosphorylated (s_1), bound and unphosphorylated (s_2), unbound and phosphorylated (s_3), and bound and phosphorylated (s_4); γ subunit: unbound and unphosphorylated (s_1), unbound and phosphorylated (s_2), and bound and phosphorylated (s_3). Internal variables v_α, v_β and v_γ record binding partners of the α, β and γ subunits, respectively. Inputs and operations (if any) are labeled together on the transition edges, separated by a delimiter symbol / (cf. Table 1). For example, $\varepsilon_1/v_\alpha := \phi$ indicates that the MFA receives a non-specific input ε_1 and then sets the variable v_α to the null value ϕ.

The machine for FcεRI has three variables, $\mathbf{v} = (v_\alpha, v_\beta, v_\gamma)$, which record the labels (id's) of binding partners for each of the corresponding domains. We note that recording binding partners using internal variables is equivalent to constructing an adjacency list to store an undirected graph. Tracking protein connectivity by such means allows protein complexes to be represented implicitly. The connectivity of proteins within a complex can be retrieved by a graph traversal. In the FcεRI pathway, some protein state transitions only happen in specific molecular contexts. For example, crosslinking of two receptors by an IgE dimer initiates signaling. On the cytoplasmic side of a crosslinked receptor dimer, a β subunit-associated Lyn can transphosphorylate the β subunit of the other receptor to initiate an intracellular signaling cascade [41]. To incorporate such non-local contextual information into the MFA-based pathway model of a signaling system, one needs to specify *reaction rules*. In the following section, we introduce a formal definition of reaction rules, which are used to describe interactions between proteins modeled by MFAs.

2.2 Molecular interactions — reaction rules

An MFA is essentially a discrete state model that characterizes a protein as a reactive agent with state transition protocols. Since specification of an MFA structure does not require consideration of the modeled protein within the larger context of a signaling system, explicit rules are needed to connect individual types of MFAs as parts of an interacting system.

A protein-protein interaction system is composed of a collection of MFAs for different types of proteins in the system. To describe the interactions between these MFAs in terms of biochemical reactions, one can specify protein interactions for the MFAs by means of reaction rules. An interaction between proteins changes the states of all participating molecules. In other words, a reaction

Table 3: State transition table for the MFA of FcϵRI.

FcϵRI.α	a	ε_1
s_1	s_2 / $v_\alpha :=$ id	–
s_2	–	s_1 / $v_\alpha := \phi$

FcϵRI.β	b	ε_2	ε_3	ε_4
s_1	$s_2/v_\beta :=$ id	–	s_3	–
s_2	–	$s_1/v_\beta := \phi$	–	–
s_3	$s_4/v_\beta :=$ id	–	–	s_1
s_4	–	$s_3/v_\beta := \phi$	–	–

FcϵRI.γ	c	ε_5	ε_6	ε_7
s_1	–	s_2	–	–
s_2	$s_3/v_\gamma :=$ id	–	s_1	–
s_3	–	–	–	$s_2/v_\gamma := \phi$

ϕ: a null symbol indicating a free site. An id is a label assigned to identify an individual MFA agent among a population of agents of one type.

synchronizes state transitions and machine reconfigurations among participant MFAs. We can view a reaction rule for an MFA-based interaction as a specification of synchronized state transitions and operations on internal variables. We formally define a reaction rule as follows.

Definition 4 (reaction rule). *A reaction rule is an injective function* $R : X \to M \backslash P$. *The sets* $X = \{x_1, x_2, ..., x_n\}$ *and* $M = \{M_1, M_2, ..., M_n\}$ *are ordered and contain inputs and MFAs, respectively.* P *is a predicate for the mapping.*

The mapping $X \to M$ simultaneously sends each individual input x_i to machine M_i and executes the machine reconfigurations if M_i is responsive to x_i. The predicate P specifies an application condition for the mapping, which usually constitutes non-local molecular contexts (or patterns). Although the number of MFAs simultaneously involved in a reaction could in principle be any finite number n, we focus on two types of elementary interactions: (1) unimolecular interactions that involve state transitions of one MFA and (2) bimolecular interactions that involve synchronized state transitions of two MFAs. Execution of a reaction rule changes the configurations of participant MFAs according to the protocols defined in the state transition tables of the MFAs. The above definition of a reaction rule requires that MFA agents be in states that can respond to the inputs. Together with the state transition tables for individual MFA types, reaction rules provide executable and programmable protocols to connect standalone MFAs into an interacting system.

In the example model of Fig. 4, all interactions in the model can be specified by four reaction rules (Table 4). Phosphorylation and dephosphorylation of automaton A are approximated as unimolecular, single-step reactions, which can be defined by two rules, $R_1 : \{\varepsilon_1\} \to \{A\}$ and $R_2 : \{\varepsilon_2\} \to \{A\}$, respectively. In these cases, a rule is merely a local pairing of a current state and an input for an MFA. The state transition and its associated operations follow the protocol defined in the state transition table. For a bimolecular association reaction between automata A and B, a reaction rule can be formulated as R_3: $\{a, a\} \to \{A, B\}$, where the MFA set and the input set are both ordered and have a one-to-one mapping. We note that the definition of a reaction rule does not

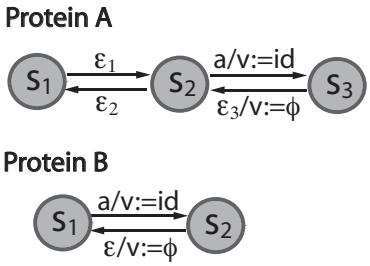


Figure 4: Interactions between MFAs. An example system that is modeled by two interacting MFAs, A and B. Automaton A models a protein that can, upon phosphorylation, bind to another protein modeled by Automaton B. Automaton A has three states: free and unphosphorylated (s_1), free and phosphorylated (s_2), and bound and phosphorylated (s_3). Automaton B has two states: free (s_1) and bound (s_2). Internal variable v's in A and B are used to record information about binding partners (i.e., the label of an MFA agent, or ϕ if free).

specify machine states and therefore only MFAs in proper states will respond to an input. A rate law specifies a mathematical formula to calculate the kinetic rate for a reaction rule, which can be used for quantitative simulations. We note that only machines in states designated by a reaction rule are accounted for when one calculates the rate according to the rate law. For example, R_3 in Table 4 specifies a mass action rate law for the association reaction between protein A and B, $r_3(t) = k_3 A(\cdot)B(\cdot)$, where $A(\cdot)$ and $B(\cdot)$ represent the eligible populations of protein A and B. Eligible machine states in an MFA can be automatically resolved by searching the state transition table for responsive states with regard to the input symbol specified in the rule. In this case, since the eligible machine states for this reaction rule are s_2 and s_1 for A and B, respectively, the actual rate should be calculated as $r_3 = k_3 A(s_2)B(s_1)$. The dissociation rule, $R_4 : \{\varepsilon_3, \varepsilon\} \rightarrow \{A, B\} \backslash A\text{–}B$, has a predicate A–B that requires A and B must share a bond. The operator '–' denotes a bond association between the two machines.

In summary, a list of reaction rules assumes three roles: (1) Assigning rate laws for quantitative computation; (2) synchronizing state transitions for bimolecular reactions; and (3) making a modeling choice to decide which subsets of machine transitions are to be included in a system, in which the specification of a set of reaction rules reflects the choice of modeling assumptions and scope. For example, some state transitions may never be triggered by a given set of reaction rules even though these transitions may be possible at the machine level.

3 Quantitative modeling

Reaction rules are essentially specifications of coupled chemical processes that can be taken to change the configuration of a system in time. A set of rules can be translated into quantitative models if the rules can be associated with rates via rate laws. A straightforward way to translate a rule-based model into a quantitative model is to automatically generate a conventional chemical

Table 4: Formal reaction rules for the model of Fig. 4

Rule description	**Formal specification**	**Rate law**
R_1 : Phosphorylation of A	$\{\varepsilon_1\} \rightarrow \{A\}$	$r_1(t) = k_1 A(\cdot)$
R_2 : Dephosphorylation of A	$\{\varepsilon_2\} \rightarrow \{A\}$	$r_2(t) = k_2 A(\cdot)$
R_3 : A and B association	$\{a, a\} \rightarrow \{A, B\}$	$r_3(t) = k_3 A(\cdot) B(\cdot)$
R_4 : A and B dissociation	$\{\varepsilon_3, \varepsilon\} \rightarrow \{A, B\} \backslash A\text{–}B$	$r_4(t) = k_4 A(\cdot)$, or $k_4 B(\cdot)$

Rules are shown as one-to-one mappings between ordered sets (e.g., $R_4 : \{\varepsilon_3, \varepsilon\} \rightarrow \{A, B\}$ indicates that ϵ_3 is an input for A and ϵ is an input for B.) A–B denotes that machines A and B have a bond association.

reaction network by evaluating reaction rules using a rewriting approach [20, 16]. However, a far more efficient approach is to use reaction rules to directly perform a simulation. Below, we describe how to construct and simulate models specified in terms of MFA structures, for either deterministic or stochastic simulation.

3.1 Deterministic simulation

A biochemical reaction system is conventionally modeled using coupled ordinary differential equations (ODEs) that describe the temporal evolution of all chemical species in the system. Here, we demonstrate that one can use a set of ODEs to instead describe the population dynamics of MFA states. In fact, an MFA state (or a combination of states) corresponds to an ensemble of chemical species, which often manifests as an experimental observable, such as free protein concentration or protein phosphorylation level. This idea is related to the concept of model reduction [19, 42, 43, 44, 45], in which a reduced system of ODEs is formulated to describe the dynamics of a set of observable quantities instead of concentrations of an exhaustive set of chemical species.

In a most general form, we can write the following ODE to model the population level of the agents of MFA M in state s, denoted as $M(s,t)$:

$$\frac{dM(s,t)}{dt} = r_{\text{in}}(t) - r_{\text{out}}(t) \ , \tag{1}$$

where r_{in} (r_{out}) is the rate of population influx (outflux), consistent with the rate laws associated with the transition rules related to state s. For a single MFA agent, the above equation describes the time rate of change of the probability to find the machine in state s.

Considering the simple example model illustrated in Fig. 4, we assume the law of mass action for protein association reactions and single-step protein phosphorylation and dephosphorylation reactions. The following differential equation can be used to describe the concentration of protein A in the machine state s_2, $A(s_2)$:

$$\begin{aligned} \frac{dA(s_2,t)}{dt} &= \underbrace{k_1 A(s_1,t) + k_4 A(s_3,t)}_{r_{\text{in}}} - \underbrace{A(s_2,t)(k_2 + k_3 B(s_1,t))}_{r_{\text{out}}} \\ &= r_1(t) + r_4(t) - (r_2(t) + r_3(t)) \ , \end{aligned} \tag{2}$$

where $X(s_i,t)$ is the concentration of protein X in its machine state s_i at time t. The parameter k_i is the rate constant for an elementary reaction process defined by Rule i, and r_i is the overall reaction rate for Rule i (Table 4). One

can systematically write down ODEs for other MFA states for the model of Fig. 4. We assume the total numbers of automata A and B are conserved, i.e., $A_{\rm tot} = A(s_1,t) + A(s_2,t) + A(s_3,t)$ and $B_{\rm tot} = B(s_1,t) + B(s_2,t)$. Note that $A(s_3,t) = B(s_2,t)$, the number of bonds formed between automata A and B. Based on these constraints, only one more independent ODE is needed to describe the whole system:

$$\begin{aligned} \frac{dA(s_3,t)}{dt} &= k_3 A(s_2,t) B(s_1,t) - k_4 A(s_3,t) \\ &= r_3(t) - r_4(t) \ . \end{aligned} \tag{3}$$

The above procedure of constructing a system of ODEs can be automated. In some systems modeled by MFAs, the construction of ODEs may not be straightforward. For example, if an MFA state transition depends on the status of a predicate evaluation, the calculation of the transition rate needs to be adjusted to account for the outcome of predicate evaluation. It is in many cases a difficult task to accurately account for rates of conditional transitions. We will discuss this issue below when we consider an example model for a MAPK cascade with a scaffold protein.

3.2 Stochastic simulation

The temporal dynamics of a biochemical reaction system can be modeled as a continuous-time discrete-state Markov process to account for the evolution of the system configuration, which can be described by the following master equation [46]:

$$\frac{dp(c,t)}{dt} = \sum_{c' \neq c} \left[w(c|c') p(c',t) - p(c,t) w(c'|c) \right] \ , \tag{4}$$

where $p(c,t)$ is the probability that the system is found in configuration c, and $w(c'|c)$ gives the transition rate from configuration c to c'. In a chemical reaction system, a configuration is defined by the concentrations of all chemical species. In a system specified by MFAs, a configuration c is determined by the states and connectivities of MFAs. More precisely, the configuration is given by the properties of the individual MFA agents in the system, including the states and internal variables of these MFA agents. Analytical solutions to the above master equation are only possible for very simple systems. For a typical system, direct numerical integrations of the master equation is often intractable because of the enormous size of the configuration space. Kinetic Monte Carlo simulation is applied to conduct sequential random walks through the configuration space and to obtain stochastic trajectories for a system of interest.

A system of rate processes can be simulated by the classic kinetic Monte Carlo method [47]. In our case, coupled processes in a biomolecular interaction system are defined by reaction rules that proceed in time at rates $\mathbf{r}$. These rule rates are determined by the current configuration of the system. At each time step, the waiting time τ for the next reaction event can be sampled from an exponential distribution with a mean waiting time of $1/r_{\rm tot}$, where $r_{\rm tot} = \sum_i r_i$ is the overall reaction rate of the system. To select a process that generates the next reaction event, one can sample a rule i proportional to its rate r_i [48].

For rule-based models defined by MFA structures, additional sampling steps are needed in each step to identify MFA agents that should undergo state transitions. Below, we outline a kinetic Monte Carlo algorithm for simulating systems specified in terms of MFAs.

[Step 1] Initialization. Set time $t = 0$, set the initial states and copy numbers of individual MFA agents, specify the rate constants of rate laws associated with reaction rules; calculate rule rates $\mathbf{r}$, and specify stopping criteria.

[Step 2] At each time step, select a rule i and a waiting time τ, and update the time $t \leftarrow t + \tau$.

[Step 3] Sample MFA agents from MFA candidates that are in permissible configurations as specified in the reaction rule sampled in Step 2, execute the state transitions of the sampled MFA agents, and recalculate the rate vector $\mathbf{r}$.

[Step 4] Repeat Steps 2 and 3 until a stopping criterion is satisfied.

In the above algorithm, Step 3 describes an agent-based simulation that tracks the states of individual proteins modeled by MFAs. A simulation produces single-protein configurations with details about reactive sites as well as connections between proteins.

Several general kinetic Monte Carlo methods for simulation of rule-based models have recently been developed [48, 49, 50, 51, 52], which can be readily adapted to suit the MFA framework. A rule-based kinetic Monte Carlo simulation involves sampling molecular agents or agent components that are permissible for transformation according to a rule [48]. In simulating an agent-based MFA model, after a rule is sampled in Step 2, the algorithm described above involves searching for reactant agents in a population of candidate MFA agents that satisfy the rule protocol. The selected rule is executed by transforming the sampled MFA agents (as in Step 3 in the above procedure). MFA transformations are executed by sending input signals as specified in the rule to the sampled MFA agents.

In some situations, the individual states of proteins and the connections of proteins within protein complexes may not be of interest, or such information may not be experimentally resolvable to verify the predictions generated by an agent-based simulation. In these cases, A kinetic Monte Carlo procedure that incorporates only observable quantities may be adopted. A kinetic Monte Carlo simulation can proceed as long as one is able to update the rates of reaction rules for each time step, which only requires tracking the populations of those machine states indicated in the rules (see Table 4). These local MFA states usually consist of experimentally accessible quantities such as the number of protein bonds or phosphorylated sites. Other quantities, such as the concentration of a complex, may in some cases be synthesized from basic MFA configurations.

4 Example: MAPK cascade

In this section, we use the example of a scaffold-mediated MAPK cascade to demonstrate how to construct and simulate an MFA-based model of a signaling pathway.

The system is inspired by the scaffold-mediated MAPK cascade in yeast. The scaffold protein Ste5 possesses three domains that bind the MAP kinases

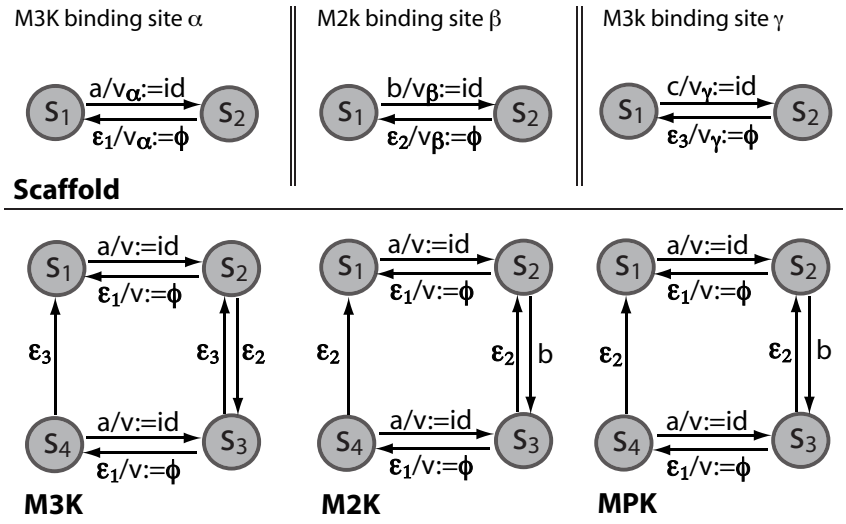


Figure 5: MFA state transition diagrams for proteins in a MAPK cascade with a scaffold protein. The scaffold protein has three submolecular binding sites: the α, β and γ sites that bind to M3K, M2K and MPK, respectively. State s_1 (s_2) for each scaffold site indicates that the site is free (bound). Each kinase has four states: s_1(free and unphosphorylated), s_2 (bound and unphosphorylated), s_3 (bound and phosphorylated), and s_4 (free and phosphorylated). Internal variables (v_α, v_β, v_γ in the scaffold, and v's in the kinases) record contextual information, in particular, record binding partners. Note that the names of inputs and internal variables are local to the MFAs in which they appear. In other words, inputs or internal variables in different MFAs with the same name are not identical.

Ste11 (MAPKKK), Ste7 (MAPKK) and Fus3 (MAPK) in the signaling pathway for the mating response [53]. We consider a scaffold protein with three independent binding sites: α, β and γ sites, and three MAP kinases: a MAPKKK that binds to the α site of the scaffold protein, a MAPKK that binds to the β site of the scaffold protein and can be phosphorylated by MAPKKK, and a MAPK that binds to the γ site of the scaffold protein and can be phosphorylated by MAPKK. We assume that (1) binding reactions of the different kinases to the scaffold protein are independent processes, (2) MAPKKK can only be phosphorylated when it is bound to the scaffold protein, (3) phosphorylation of MAPKK (MAPK) can happen only when its kinase, phosphorylated MAPKKK (MAPKK), colocates on the same scaffold protein, and (4) phosphorylation and dephosphorylation of kinases can be modeled as single-step processes.

To simplify notations, we will use SCF to denote the scaffold protein and M3K, M2K and MPK to denote MAPKKK, MAPKK and MAPK, respectively. Figure 5 illustrates state transition diagrams of MFA models for the four proteins involved in the system. A total of 12 reaction rules describing protein interactions in the system are listed in Table 5.

In particular, we note that assumption (3) above requires checking molecular contexts to determine if protein state transitions are permissible, which exemplifies one important feature of MFA framework that allows modeling context-

sensitive interactions. Consequently, Rules 9 and 11 in Table 5 require predicate evaluations that examine nonlocal machine configurations. For example, Rule 9 validates an M2K agent by checking whether it is bound to an SCF agent that in turn is bound to an M3K agent in s_3. This contextual information is provided in the rule as a pattern of bond associations, M2K–SCF–M3K(s_3). In general, a pattern of a multiprotein complex can be specified by a data structure that represents a connectivity graph.

4.1 Deterministic ODEs

We first show how to write deterministic ODEs to describe the evolution of the population levels of MFA states. Our MFA model of the MAPK cascade consists of 15 independent machine states, compared to a total 33 distinct chemical species implied by the same set of reaction rules. Furthermore, we assume that the total amount of each protein is conserved, which corresponds to $M_{\rm tot} = \sum_{i=1}^{n} M(s_i)$ for an MFA with n states, where $M_{\rm tot}$ is the total number of the MFA agents. Kinetic parameters that appear in the following equations are defined in Table 5. The following equation characterizes M3K binding to the α site of the scaffold protein:

$$\frac{d\mathrm{SCF}.\alpha(s_1)}{dt} = -k_1[\mathrm{M3K}(s_1) + \mathrm{M3K}(s_4)]\mathrm{SCF}.\alpha(s_1) + k_2\mathrm{SCF}.\alpha(s_2)\ . \qquad (5)$$

The equations for M2K and MPK binding to the β and γ sites of the scaffold are similar. Together with two algebraic relationships, $\mathrm{M3K_{tot}} = \sum_{i=1}^{4} \mathrm{M3K}(s_i)$ and $\mathrm{SCF}.\alpha(s_2) = \mathrm{M3K}(s_2) + \mathrm{M3K}(s_3)$, two additional ODEs are needed to completely account for the populations of the four possible states of machine M3K:

$$\frac{d\mathrm{M3K}(s_1)}{dt} = k_2\mathrm{M3K}(s_2) + k_8\mathrm{M3K}(s_4) - k_1\mathrm{SCF}.\alpha(s_1)\mathrm{M3K}(s_1) \qquad (6)$$

$$\frac{d\mathrm{M3K}(s_2)}{dt} = k_1\mathrm{SCF}.\alpha(s_1)\mathrm{M3K}(s_1) + k_8\mathrm{M3K}(s_3) - (k_2 + k_7)\mathrm{M3K}(s_2) \qquad (7)$$

For the case of M2K or MPK, because phosphorylation of a kinase bound to the scaffold (transition from state s_2 to s_3) is a conditional process that requires colocalization of an upstream kinase on the same scaffold protein, only a fraction of all kinases in state s_2 are candidates for transition to s_3. One can use the MFA state transition diagrams of Fig. 5 to write the ODEs that track the populations of states s_1 and s_2 for M2K as follows:

$$\frac{d\mathrm{M2K}(s_1)}{dt} = k_4\mathrm{M2K}(s_2) + k_{10}\mathrm{M2K}(s_4) - k_3\mathrm{SCF}.\beta(s_1)\mathrm{M2K}(s_1) \qquad (8)$$

$$\frac{d\mathrm{M2K}(s_2)}{dt} = k_3\mathrm{SCF}.\beta(s_1)\mathrm{M2K}(s_1) + k_{10}\mathrm{M2K}(s_3) - (k_4 + k_9\theta_1)\mathrm{M2K}(s_2) \qquad (9)$$

Similar ODEs can be written to describe the dynamics of MPK states. The handling of Rule 9 (11) in Table 5 for M2K (MPK) phosphorylation requires special attention. Not all M2Ks (or MPKs) in state s_2 are subject to transition to s_3 at a given time. The eligible fraction must satisfy the model assumption that requires colocalization of an M2K (MPK) with its enzyme, a phosphorylated M3K (M2K), on the same scaffold protein. The factors θ_1 and θ_2 are

Table 5: Reaction rules for the MAPK cascade with a scaffold protein

Rule description	Formal specification	Rate law
1. M3K recruitment	$\{a, a\}\rightarrow\{\text{SCF}.\alpha,\text{M3K}\}$	$k_1\text{M3K}(\cdot)\text{SCF}.\alpha(\cdot)$
2. M3K dissociation	$\{\varepsilon_1, \varepsilon_1\}\rightarrow\{\text{SCF}.\alpha,\text{M3K}\}\backslash\text{SCF}.\alpha$–M3K	$k_2\text{SCF}.\alpha(\cdot)$
3. M2K recruitment	$\{b, a\}\rightarrow\{\text{SCF}.\beta,\text{M2K}\}$	$k_3\text{M2K}(\cdot)\text{SCF}.\beta(\cdot)$
4. M2K dissociation	$\{\varepsilon_2, \varepsilon_1\}\rightarrow\{\text{SCF}.\beta,\text{M2K}\}\backslash\text{SCF}.\beta$–M2K	$k_4\text{SCF}.\beta(\cdot)$
5. MPK recruitment	$\{c, a\}\rightarrow\{\text{SCF}.\gamma,\text{MPK}\}$	$k_5\text{MPK}(\cdot)\text{SCF}.\gamma(\cdot)$
6. MPK dissociation	$\{\varepsilon_3, \varepsilon_1\}\rightarrow\{\text{SCF}.\gamma,\text{MPK}\}\backslash\text{SCF}.\gamma$–MPK	$k_6\text{SCF}.\gamma(\cdot)$
7. M3K phosphorylation	$\{\varepsilon_2\}\rightarrow\{\text{M3K}\}$	$k_7\text{M3K}(\cdot)$
8. M3K dephosphorylation	$\{\varepsilon_3\}\rightarrow\{\text{M3K}\}$	$k_8\text{M3K}(\cdot)$
9. M2K phosphorylation	$\{b\}\rightarrow\{\text{M2K}\}\backslash\text{M2K–SCF–M3K}(s_3)$	$k_9\text{M2K}(\cdot)$
10. M2K dephosphorylation	$\{\varepsilon_2\}\rightarrow\{\text{M2K}\}$	$k_{10}\text{M2K}(\cdot)$
11. MPK phosphorylation	$\{b\}\rightarrow\{\text{MPK}\}\backslash\text{MPK–SCF–M2K}(s_3)$	$k_{11}\text{MPK}(\cdot)$
12. MPK dephosphorylation	$\{\varepsilon_2\}\rightarrow\{\text{MPK}\}$	$k_{12}\text{MPK}(\cdot)$

introduced to account for the fractions of $\text{M2K}(s_2)$ and $\text{MPK}(s_2)$ that are eligible for transition to state s_3. In general, it is non-trivial to obtain analytical equations for factors such as θ_1 and θ_2. In this example, we simply approximate these factors as follows: $\theta_1 \approx \text{M3K}(s_3)/\text{SCF}_{\text{tot}}$ and $\theta_2 \approx \text{M2K}(s_3)/\text{SCF}_{\text{tot}}$. These approximations are exact only if kinase phosphorylation reactions are independent and context insensitive. Figure 6 shows results from deterministic ODE simulations, compared to those from kinetic Monte Carlo simulations (performed as described below). We note that the stochastic simulation results are exact. The time trajectories for phosphorylated M3K from the deterministic and the stochastic simulations agree with each other on average. However, the ODE-based simulations of M2K and MPK phosphorylation deviate from those of the exact stochastic simulations due to the approximations used for θ_1 and θ_2 (Fig. 6(b)). Writing ODEs to directly describe MFA states as variables provides a way of quickly constructing a quantitative (sometimes approximate) model for simulation. One can avoid using approximations by expanding reaction rules into a chemical reaction network [16, 20]. In such cases, the variables in these ODEs would correspond to concentrations of chemical species rather than MFA states. Generating these ODEs may be impractical, because the number of ODEs needed to capture the dynamics of the chemical species implied by a set of MFAs can be much larger than the number of ODEs needed to capture the dynamics of MFA states.

4.2 Kinetic Monte Carlo simulation

The stochastic simulations were carried out using the kinetic Monte Carlo algorithm described in the previous section. The system-specific implementation was an agent-based simulation procedure that samples the reaction rule list in Table 5 and transforms individual MFA agents. We note that $\text{M2K}_\theta(s_2)$ and $\text{MPK}_\theta(s_2)$ in Rules 9 and 11 (Table 5), in contrast to the approximations used in the ODE simulations, are updated by on-the-fly bookkeeping that tracks reaction events immediately coupled to the two rules. The bookkeeping accounts for the numbers of M2K and MPK kinases in state s_2 that are eligible for the transitions specified by Rules 9 and 11. This implementation corresponds to a rejection-free algorithm for kinetic Monte Carlo simulation of rule-based models [49, 52], in which the exact rates of rules are calculated. This scheme becomes difficult to implement and computationally inefficient when predicates

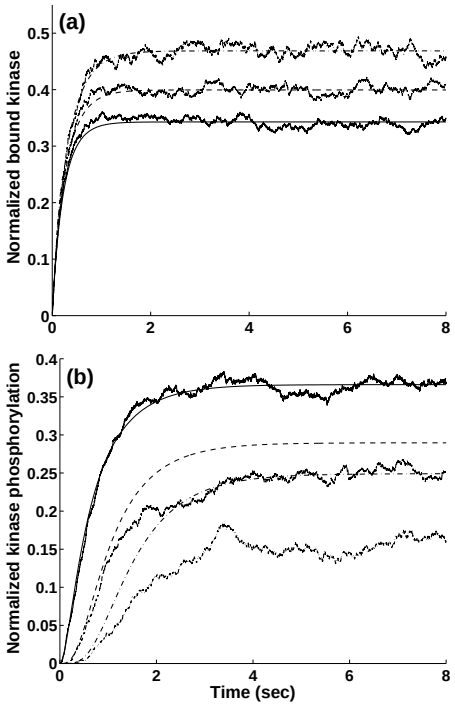


Figure 6: Simulation of the model for the MAP kinase cascade with a scaffold of Fig. 5. Results from ODE simulations (smooth curves) vs. those from kinetic Monte Carlo simulations (fluctuating curves). (a) Kinases bound to the scaffold M3K($s_2 + s_3$) (solid line), M2K($s_2 + s_3$) (dashed line) and MPK($s_2 + s_3$) (dash-dot) each normalized by total number of kinases. (b) Phosphorylated kinases M3K($s_3 + s_4$) (solid line), M2K($s_3 + s_4$) (dashed line) and MPK($s_3 + s_4$) (dash-dot) each normalized by total number of kinases. $\mathrm{SCF_{tot}} = 1000$, $\mathrm{M3K_{tot}} = 2000$, $\mathrm{M2K_{tot}} = 1500$ and $\mathrm{MPK_{tot}} = 1000$. Initially, all machines are in state s_1. Kinetic parameters (k_1 to k_{12}, defined in Table 5) are chosen for the purpose of illustration, not to model a specific system. $k_1 = k_3 = k_5 = 1.66 \times 10^{-6}$ nL·s^{-1}, $k_2 = k_4 = k_6 = 1.0$ s^{-1}, $k_7 = k_9 = k_{11} = 3.0$ s^{-1} and $k_8 = k_{10} = k_{12} = 1.0$ s^{-1}.

can be potentially affected by many types of reaction events. For example, Rule 9 may be affected by events from eight other distinct rules, as indicated in the dependence matrix $\mathcal{D}$ below. When the algorithm executes an event defined by any of these eight rules, the algorithm also needs to evaluate whether the predicate of Rule 9 is affected.

Reaction rules are usually coupled in most systems. In other words, an event from one rule may affect the rates of others. Such coupling relationships between rules can be summarized by a "dependency graph," or "influence map" [13]. For the example of the MAPK cascade model, we can summarize

the rule dependence in the form of the following adjacency matrix:

$$\mathcal{D} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & x & 0 & x & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & x & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & x & 0 & x & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & x & 0 & x & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & x & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x & 1 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & 1 \end{bmatrix} . \tag{10}$$

A matrix entry d_{ij} at row i and column j represents the influence of an event from rule i on the rate of rule j. An entry with a boolean value 1 (0) indicates that an event in a rule has (does not have) an immediate impact on the rate of another rule, whereas x indicates that an event may or may not alter the rate of another rule, depending on whether the event changes the predicate of the other rule. For the example of the MAPK cascade, an event from Rule 1 can alter the rate of Rule 9 only in the case that a recruited M3K is phosphorylated and an unphosphorylated M2K is bound to the same scaffold agent. A dependency graph can in general be automatically derived by systematically analyzing a reaction rule set. In practice, the adjacency matrix $\mathcal{D}$ can be made more quantitative by replacing the Boolean value 1's with pre-calculated values of rate changes. For example, the entry d_{89} (the influence of an event from Rule 8 on the rate of Rule 9) can be (conditionally) replaced by a numerical value, the value of $-k_9$ (the minus sign indicates a reduction in the rule rate), because an M3K dephosphorylation may decrement the eligible population of M2K agents.

To reduce bookkeeping, one may use a rejection algorithm [48] as an alternative. This algorithm allows one to use rejection sampling to simplify the firing of reactions when state transitions are associated with predicate functions. In this algorithm, one calculates rule rates and samples participant MFA agents without considering constraints imposed by the predicate functions. Sampled trial agents are rejected for state transition if the predicate function does not evaluate as true. For example, the number of all MPK agents in state s_2 can be used to calculate the rate of Rule 11 as $\widetilde{r}_{11} = k_{11}\mathrm{MPK}(s_2)$. Once an event from Rule 11 is sampled, a trial MPK agent in state s_2 will be chosen, which will then undergo a state transition only if the transition condition is satisfied. Otherwise, the transition is rejected. Implementation of a rejection algorithm is easier compared to that of a rejection-free algorithm. The rejection algorithm enforces the conditions of state transitions only when a rule that has conditional transitions is sampled. Our experience suggests that a rejection algorithm is more computationally efficient than a rejection-free algorithm as long as rejections do not comprise the vast majority of all Monte Carlo steps [49].

The sparsity of dependency matrix $\mathcal{D}$ in Eq. (10) (with many entries being zeros) indicates that rule couplings are largely localized. Similar to an efficient implementation of a conventional stochastic simulation algorithm for chemical reaction systems [54], for a large system specified by a considerable number of

rules, weak dependency between rules can be used to optimize the procedure for updating rule rates after each Monte Carlo step.

5 Discussion

Information processing in living cells can be viewed as protein computation, in which proteins act as computing machines that react to external signals by local computation [31, 55]. Thus, a protein-protein interaction system is an integrative and distributed system with numerous computing devices interacting with each other under certain protocols. This perspective suggests that formal computing models can help archive, organize and interpret protein functions and their interactions. Formal structures also facilitate the process of computational modeling of complex and large-scale protein-protein interaction systems.

In this work, we introduce an extension to the traditional computing model of finite state automata to describe protein behaviors in response to external inputs and protein interactions. The MFA formalism offers a rule-based platform for modeling and simulating biochemical systems, especially for signal transduction. An MFA is in essence a data structure that specifies protocols to define the activity of a protein in a discrete state space. An MFA can be used as a representation of knowledge or hypotheses about a protein and can serve as a building block for biomolecular interaction models. At the systems level, reaction rules connect separate MFA-represented proteins and model the dynamics of a protein interaction system as synchronized MFA state transitions. The MFA formalism allows for a clearer and more natural representation of proteins and the combinatorics of protein interactions. The MFA formalism adds in intuitive formulation of mechanistic models of signal transduction, which can be accessible for those who are familiar with the biological knowledge underlying the models. For quantitative model computation, as our example model of a MAP kinase cascade system demonstrates, ODEs for deterministic simulations can be constructed to track concentrations of machine states that often directly correspond to experimentally resolvable quantities. (For some states, the ODE solutions give approximations.) For exact and stochastic simulations, rule-based kinetic Monte Carlo methods can be applied.

Formalisms derived from finite automata theory have been proposed earlier for applications in biology. Notably, Harel and coworkers [56] have applied statecharts, a graphical and hierarchical (extended) finite automata structure [50], to model and simulate cell development and dynamics of cell populations. Recently, use of finite automata to model biomolecular interactions was studied by Cardelli [57], in which the author introduced the concept of polyautomata. The concepts of polyautomata and MFA are closely related. In Ref. [57], polyautomata are used to represent SPiM scripts, which specify stochastic simulations via the stochastic pi-calculus approach implemented in the SPiM software tool [58]. Here, we have shown that MFAs can be used to specify both deterministic and stochastic simulation approaches (Fig. 6). Cardelli [57] demonstrated that polyautomata are useful for modeling protein complex formation, including polymerization-like reactions. Here, we have shown that MFAs can be used to model protein complex formation as well as post-translational modifications of proteins, as illustrated in Figs. 3 and 5. Finally, the MFA formalism extends the polyautomata concept in an important way by allowing for the explicit represen-

tation of the functional components of proteins and the structural relationships among these components (Fig. 2).

A potential strength of the MFA framework is the mature development and many applications of finite automata. Various forms of finite state automata have been industry standards for modeling reactive systems for many years. In particular, MFAs are amenable to hardware implementations using programmable logic devices including widely-used programmable array logic (PAL), generic array logic (GAL) and field programmable gate array (FPGA) devices. In particular, because of the sequential nature of discrete event-driven simulation, the performance of simulating large-scale complex biochemical reaction systems by stochastic simulation is poor even if it is not prohibitive. Hardware implementation may yield an advantage in speed. The first hardware (FPGA) stochastic simulations of biochemical networks were implemented by Keane and co-workers [59] and Salwinski and Eisenberg [60]. Taking advantage of the parallel architecture of FPGA, implementations by Salwinski and Eisenberg [60] allowed improvement in the speed of simulation of a simple bimolecular reaction by at least one order of magnitude compared to a conventional software implementation on a benchmarking platform. Recently, Yoshimi et al. [61] implemented the next reaction method of Gibson and Bruck [54] in an FPGA-based simulator and were able to achieve a significant speed-up. Implementation of rule-based models into programmable circuits has yet to be realized, but finite state machines are routinely implemented in hardware. Hardware implementation stores state variables and embeds state transition protocols into digital electronic circuits. In the design process, the MFA structures need to be translated into binary logic to apply digital computing to achieve the defined machine dynamics. As proteins can be modeled as MFAs, in principle, the dynamics of a protein interaction network may be simulated by electronic circuits.

Another potential use of the MFA structure is to archive proteins in terms of their reaction dynamics. Since an MFA is a standalone structure for storing the discrete dynamics of a protein, we expect that it can be used to systematically archive protein functions, with the MFA structure serving as an elementary record type for a database. Protein records in current protein databases are mostly annotations including amino acid sequences, functional domains, associated functions, etc. However, such information cannot be readily used to construct mechanistic biomolecular interaction models. The MFA structure offers an alternative way of storing protein records. Using a database with MFA records, one can construct a model by querying the database for MFA structures to obtain a set of desired molecular building blocks. One can then specify reaction rules to connect these machines. Such a rule-based model can be efficiently revised and incrementally refined when records of MFAs involved in the model are updated to reflect new knowledge.

Acknowledgment

This work was supported by grants GM076570, GM085273 and RR18754 from the National Institutes of Health, by the Department of Energy through contract DE-AC52-06NA25396, and by grant 30870477 from the National Science Foundation of China to J.Y. We thank the Center for Nonlinear Studies for support that made it possible for J.Y. to visit Los Alamos.

References

[1] Kitano H.: 'Computational systems biology', *Nature*, 2002, 420, pp. 206–210

[2] Kholodenko B.N.: 'Cell signalling dynamics in time and space', *Nat. Rev. Mol. Cell Biol.*, 2006, 7, pp. 165–176

[3] Aldridge B.B., Burke J.M., Lauffenburger D.A., Sorger P.K.: 'Physicochemical modelling of cell signalling pathways', *Nat. Cell Biol.*, 2006, 8, pp. 1195–1203

[4] Breitling R., Hoeller D.: 'Current challenges in quantitative modeling of epidermal growth factor signaling', *FEBS Lett.*, 2005, 579, pp. 6289–6294

[5] Hlavacek W.S., Faeder J.R., Blinov M.L., Perelson A.S., Goldstein B.: 'The complexity of complexes in signal transduction', *Biotechnol. Bioeng.*, 2003, 84, pp. 783–794

[6] Yang X.J.: 'Multisite protein modification and intramolecular signaling', *Oncogene*, 2005, 24, pp. 1653–1662

[7] Hunter T.: 'Signaling—2000 and beyond', *Cell*, 2000, 100, pp. 113–127

[8] Pawson T., Nash P.: 'Assembly of cell regulatory systems through protein interaction domains', *Science*, 2003, 300, pp. 445–452

[9] Bhattacharyya R.P., Reményi A., Yeh B.J., Lim W.A.: 'Domains, motifs, and scaffolds: The role of modular interactions in the evolution and wiring of cell signaling circuits', *Annu. Rev. Biochem.*, 2006, 75, pp. 655–80

[10] Hlavacek W.S., Faeder J.R.: 'The complexity of cell signaling and the need for a new mechanics', *Sci. Signal.*, 2009, 2, p. pe46

[11] Mayer B.J., Blinov M.L., Loew L.M.: 'Molecular machines or pleiomorphic ensembles: signaling complexes revisited', *J. Biol.*, 2009, 8, p. 81

[12] Hlavacek W.S., Faeder J.R., Blinov M.L., Posner R.G., Hucka M., Fontana W.: 'Rules for modeling signal-transduction systems', *Sci. STKE*, 2006, 2006:re6

[13] Danos V., Feret J., Fontana W., Harmer R., Krivine J.: 'Rule-based modelling of cellular signalling', *Lect. Notes Comput. Sci.*, 2007, 4703, pp. 17–41

[14] Danos V., Laneve C.: 'Formal molecular biology', *Theor. Comput. Sci.*, 2004, 325, pp. 69–110

[15] Priami C., Quaglia P.: 'Beta binders for biological interactions', *Lect Notes Comput. Sci.*, 2005, pp. 20–33

[16] Faeder J.R., Blinov M.L., Goldstein B., Hlavacek W.S.: 'Rule-based modeling of biochemical networks', *Complexity*, 2005, 10, pp. 22–41

[17] Blinov M.L., Yang J., Faeder J.R., Hlavacek W.S.: 'Graph theory for rule-based modeling of biochemical networks', *Lect. Notes Comput. Sci.*, 2006, 4230, pp. 89–106

[18] Andrei O., Kirchner H.: 'Graph rewriting and strategies for modeling biochemical networks', in Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (IEEE Computer Society, 2007), pp. 407–414

[19] Feret J., Danos V., Krivine J., Harmer R., Fontana W.: 'Internal coarse-graining of molecular systems', *Proc. Natl. Acad. Sci. USA*, 2009, 106, pp. 6453–6458

[20] Blinov M.L., Faeder J.R., Goldstein B., Hlavacek W.S.: 'BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains', *Bioinformatics*, 2004, 20, pp. 3289–3291

[21] Moraru I.I., Schaff J.C., Slepchenko B.M., Blinov M.L., Morgan F., Lakshminarayana A., Gao F., Li Y., Loew L.M.: 'Virtual Cell modelling and simulation software environment', *IET Syst. Biol.*, 2008, 2, pp. 352–362

[22] Faeder J.R., Blinov M.L., Hlavacek W.S.: 'Rule-based modeling of biochemical systems with BioNetGen', *Methods Mol. Biol*, 2009, 500, pp. 113–167

[23] Mallavarapu A., Thomson M., Ullian B., Gunawardena J.: 'Programming with models: modularity and abstraction provide powerful capabilities for systems biology', *J. Roy. Soc. Interface*, 2009, 6, pp. 257–270

[24] Lok L., Brent R.: 'Automatic generation of cellular reaction networks with moleculizer 1.0', *Nat. Biotechnol.*, 2005, 23, pp. 131–136
[25] Andrews S.S., Addy N.J., Brent R., Arkin A.P., Sauro H.M.: 'Detailed simulations of cell biology with Smoldyn 2.1', *PLoS Comput. Biol.*, 2010, 6, p. e1000705
[26] Meier-Schellersheim M., Xu X., Angermann B., Kunkel E.J., Jin T., Germain R.N.: 'Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method', *PLoS Comput. Biol.*, 2006, 2, p. e82
[27] Lis M., Artyomov M.N., Devadas S., Chakraborty A.K.: 'Efficient stochastic simulation of reaction-diffusion processes via direct compilation', *Bioinformatics*, 2009, 25, pp. 2289–2291
[28] Morton-Firth C.J., Bray D.: 'Predicting temporal fluctuations in an intracellular signalling pathway', *J. Theor. Biol.*, 1998, 192, pp. 117–128
[29] Regev A., Silverman W., Shapiro E.: 'Representation and simulation of biochemical processes using the π-calculus process algebra', *Pacific Symposium on Biocomputing*, 2001, 6, pp. 459–470
[30] Priami C., Regev A., Shapiro E., Silverman W.: 'Application of a stochastic name-passing calculus to representation and simulation of molecular processes', *Inform. Process. Lett.*, 2001, 80, pp. 25–31
[31] Regev A., Shapiro E.: 'Cellular abstractions: Cells as computation', *Nature*, 2002, 419, p. 343
[32] Fisher J., Henzinger T.A.: 'Executable cell biology', *Nat. Biotechnol.*, 2007, 25, pp. 1239–1250
[33] Gillespie D.T.: 'Stochastic simulation of chemical kinetics', *Annu. Rev. Phys. Chem*, 2007, 58, pp. 35–55
[34] Hopcroft J.E., Motwani R., Ullman J.D.: Introduction to Automata Theory, Languages, and Computation, 2nd. edition (Addison-Wesley, New York, 2000)
[35] Sipser M.: Introduction to the Theory of Computation, 2nd edition (Course Technology, 2005)
[36] Lee D., Yannakakis M.: 'Principles and methods of testing finite state machines — a survey.', *Proc. IEEE*, 1996, 84, pp. 1090–1123
[37] Börger E., Stärk R.F.: Abstract State Machines: A Method for High-Level System Design and Analysis (Springer Verlag, New York, 2003)
[38] Mohri M.: 'Finite-state transducers in language and speech processing', *Computational Linguistics*, 1997, 23, pp. 269–311
[39] Brand D., Zafiropulo P.: 'On communicating finite-state machines', *J. ACM*, 1983, 30, pp. 323–342
[40] Goldstein B., Faeder J.R., Hlavacek W.S., Blinov M.L., Redondo A., Wofsy C.: 'Modeling the early signaling events mediated by FcεRI', *Mol. Immunol.*, 2002, 38, pp. 1213–1219
[41] Faeder J.R., Hlavacek W.S., Reischl I., Blinov M.L., Metzger H., Redondo A., Wofsy C., Goldstein B.: 'Investigation of early events in FcϵRI-mediated signaling using a detailed mathematical model.', *J. Immunol.*, 2003, 170, pp. 3769–3781
[42] Borisov N.M., Markevich N.I., Hoek J.B., Kholodenko B.N.: 'Signaling through receptors and scaffolds: independent interactions reduce combinatorial complexity', *Biophys. J.*, 2005, 89, pp. 951–966
[43] Conzelmann H., Saez-Rodriguez J., Sauter T., Kholodenko B.N., Gilles E.: 'A domain-oriented approach to the reduction of combinatorial complexity in signal transduction networks', *BMC Bioinformatics*, 2006, 7, p. 34
[44] Borisov N.M., Markevich N.I., Hoek J.B., Kholodenko B.N.: 'Trading the micro-world of combinatorial complexity for the macro-world of protein interaction domains', *BioSystems*, 2006, 83, pp. 152–166
[45] Borisov N.M., Chistopolsky A.S., Faeder J.R., Kholodenko B.N.: 'Domain-oriented reduction of rule-based network models', *IET Syst. Biol*, 2008, 2, pp. 342–351

[46] van Kampen N.G.: Stochastic processes in physics and chemistry, 3rd edition (North-Holland, Boston, 2007)

[47] Voter A.F.: 'Introduction to the kinetic Monte Carlo method', *Radiation Effects in Solids*, 2007, 235, pp. 1–23

[48] Yang J., Monine M.I., Faeder J.R., Hlavacek W.S.: 'Kinetic Monte Carlo method for rule-based modeling of biochemical networks', *Phys. Rev. E*, 2008, 78, p. 031910

[49] Yang J., Hlavacek W.S.: 'Rejection-free kinetic Monte Carlo simulation of multivalent biomolecular interactions', *Arxiv preprint arXiv:0812.4619v5*, 2010

[50] Danos V., Feret J., Fontana W., Krivine J.: 'Scalable simulation of cellular signaling networks', *Lect. Notes Comput. Sci.*, 2007, 4807, pp. 139–157

[51] Colvin J., Monine M.I., Faeder J.R., Hlavacek W.S., Von Hoff D.D., Posner R.G.: 'Simulation of large-scale rule-based models', *Bioinformatics*, 2009, 25, pp. 910–917

[52] Colvin J., Monine M.I., Gutenkunst R.N., Hlavacek W.S., Von Hoff D.D., Posner R.G.: 'RuleMonkey: software for stochastic simulation of rule-based models', *BMC Bioinformatics*, 2010, 11, p. 404

[53] Garrington T.P., Johnson G.L.: 'Organization and regulation of mitogen-activated protein kinase signaling pathways', *Curr. Opin. Cell Biol.*, 1999, 11, pp. 211–218

[54] Gibson M.A., Bruck J.: 'Efficient exact stochastic simulation of chemical systems with many species and many channels', *J. Phys. Chem. A*, 2000, 104, pp. 1876–1889

[55] Bray D.: 'Protein molecules as computational elements in living cells', *Nature*, 1995, 376, pp. 307–312

[56] Fisher J., Harel D.: 'On statecharts for biology', in Symbolic systems biology: theory and methods (Jones and Bartlett, 2010, in press)

[57] Cardelli L.: 'Artificial biochemistry', in A. Condon, D. Harel, J.N. Kok, A. Salomaa, E. Winfree (Eds.), Algorithmic Bioprocesses, Natural Computing Series (Springer, Berlin, 2009), pp. 429–462

[58] Phillips A., Cardelli L.: 'Efficient, correct simulation of biological processes in the stochastic pi-calculus', *Lect. Notes Comput. Sci.*, 2007, 4695, pp. 184–199

[59] Keane J.F., Bradley C., Ebeling C.: 'A compiled accelerator for biological cell signaling simulations', in Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays (ACM, New York, 2004), pp. 233–241

[60] Salwinski L., Eisenberg D.: 'In silico simulation of biological network dynamics', *Nat. Biotechnol.*, 2004, 22, pp. 1017–1019

[61] Yoshimi M., Iwaoka Y., Nishikawa Y., Kojima T., Osana Y., Funahashi A., Hiroi N., Shibata Y., Iwanaga N., Yamada H., Kitano H., Amano H.: 'FPGA implementation of a data-driven stochastic biochemical simulator with the next reaction method', in International Conference on Field Programmable Logic and Applications (IEEE, 2007), pp. 254–259